GYMNASIUM OTTOBRUNN

Oberstufenjahrgang 2017/19

Seminarfach Informatik

Seminararbeit

Kalender auf Basis von XML

Verfasser: Eric Brendel

Seminarleiter: Peter Brichzin

Bewertung: …………. Punkte

Unterschrift des Seminarleiters: …………………………..

Inhaltsverzeichnis

[Vorwort 1](#_Toc527473307)

[Einleitung 2](#_Toc527473308)

[XML und DTD 2](#_Toc527473309)

[XML 2](#_Toc527473310)

[Was bedeutet das für die Implementierung? 3](#_Toc527473311)

[Welchen Nutzen kann der Programmierer daraus ziehen? 3](#_Toc527473312)

[DTD 3](#_Toc527473313)

[Notwendigkeit und Vorteile einer DTD 3](#_Toc527473314)

[Beispiel einer DTD (Auszug aus der CalML) 4](#_Toc527473315)

[Terminkalender 5](#_Toc527473316)

[Produkt und Funktionen 5](#_Toc527473317)

[Beginn des Projekts 5](#_Toc527473318)

[User Stories 5](#_Toc527473319)

[Tasks und Prototypen 6](#_Toc527473320)

[Zeitliche Planung 6](#_Toc527473321)

[Zusammenarbeit der Klassen (Klassendiagramm) 6](#_Toc527473322)

[DTD (eigene Sprache) 6](#_Toc527473323)

[Schnittstelle Java XML (Datenspeicherung) 6](#_Toc527473324)

[Wahl der verwendeten Bibliothek 6](#_Toc527473325)

[Implementierung 7](#_Toc527473326)

[Logic 8](#_Toc527473327)

[GUI mit JavaFX 8](#_Toc527473328)

[Entscheidung über Methode zur Erstellung der GUI 8](#_Toc527473329)

[Implementierung 9](#_Toc527473330)

[Einschätzung des Projekts (Fazit) 9](#_Toc527473331)

[Einschätzung des Projekts 9](#_Toc527473332)

[Einschätzung des Produkts 9](#_Toc527473333)

[Literaturverzeichnis 9](#_Toc527473334)

# 

# Vorwort

Diese Seminararbeit beschreibt das Entstehen und die Abläufe eines in Java geschriebenen Programms. Um das Geschriebene verstehen zu können, ist es daher notwendig, Grundkenntnisse in Java oder anderen objektorientierten Programmiersprachen zu haben. Vorkenntnisse zum Thema XML sind nicht notwendig, da dieses Thema vor der eigentlichen Erläuterung des Projekts erklärt wird.

# Einleitung

In unserer heutigen Zeit wird innerhalb der Familie alles immer hektischer. Das eine Kind muss um 15.00 Uhr zum Fußball, das andere zum Tennis und Mama hat sich abends zum Essen mit Freundinnen verabredet. Wenn solche Situationen alltäglich sind, dann kann es schon einmal zu Überschneidungen kommen. Kind A will sich mit Freunden treffen, dabei hat es aber schon einen Arzttermin, von dem es gar nichts weiß. Die Familie will einen Ausflug machen, dabei hat das eine Kind aber am Tag darauf eine wichtige Matheklausur und noch gar nichts gelernt. So etwas kommt leider relativ häufig vor. Ein einfacher Kalender, der in der Küche oder im Wohnzimmer hängt, ist da nicht mehr der beste Weg. Es muss eine flexiblere Lösung her.



In dieser Seminararbeit wird der Entwicklungsprozess eines solchen digitalen Familienkalenders beschrieben. Dieser Kalender unterscheidet sich zudem durch einige praktische Funktionen, die der Familienfreundlichkeit dienen sollen, von einem herkömmlichen Kalender.

Es wird erläutert, wie man die gesammelten Daten speichert, wieder ausliest, verarbeitet und letztendlich visualisiert.

# XML und DTD

Um einen guten Einstieg in die Thematik zu liefern und um das Verständnis für die folgende Seminararbeit zu verbessern, behandelt dieser kurze Abschnitt die Themen XML und DTD, die einen wichtigen Grundpfeiler für das mit der Arbeit zusammenhängende Projekt darstellen.

## XML

Bei XML handelt es sich um die sogenannte „eXtensible Markup-Language“[[1]](#footnote-1). Bei einer Markupsprach handelt es sich nicht um eine Programmiersprache, wie beispielsweise Java, C++ oder Python, sondern um eine Auszeichnungssprache. Das „eXtensible“ ist Englisch und bedeutet „erweiterbar“. Es ist also möglich, die Sprache zu erweitern, bzw. eine neue Sprache zu erstellen und sie an die eigenen Ansprüche anzupassen.

### Was bedeutet das für die Implementierung?

Der wesentliche Unterschied zwischen einer Programmier- und eine Markup-/ Auszeichnungssprache besteht darin, dass es sich bei Letzterer um eine beschreibende Sprache handelt. Ein weiterer Unterschied besteht jedoch auch darin, dass mit Markupsprachen Dokumente erzeugt werden und mit Programmiersprachen Programme.   
Für den Programmierer bedeutet das im Wesentlichen, dass eine Auszeichnungssprache die Struktur von Daten beschreibt und Programmiersprachen für die Realisierung logischer Abläufe verwendet werden.   
Dies ist zum Beispiel schlicht daran erkennbar, dass es in Auszeichnungssprachen weder bedingte Anweisungen, Wiederholungsanweisungen oder Methoden gibt, sondern nur Elemente und deren Attribute, beziehungsweise ihren Inhalt[[2]](#footnote-2).

### Welchen Nutzen kann der Programmierer daraus ziehen?

XML ist ein sehr einfacher und praktischer Weg, Daten persistent zu speichern. Man kann mit Hilfe von Bibliotheken sogar ganze Objektzustände ohne größeren Aufwand speichern oder auslesen. Ein Vorteil ist hierbei auch, dass die Dateien nicht an eine Programmiersprache oder ein Betriebssystem gebunden, sondern von allen Betriebssystemen und Programmiersprachen benutzt werden können. Hierfür ist jedoch eine entsprechende Bibliothek notwendig. Eine sehr ähnliche Sprache, „HTML“, die als Vorbild für XML diente, wird zum Beispiel zum Darstellen von Webseiten genutzt. FXML, eine mit XML erstellte Sprache, wird verwendet, um Java-Applikationen eine Benutzeroberfläche zu geben und somit die Nutzerinteraktion einfacher zu gestalten. Auch in diesem Projekt ist die GUI mit JavaFX und FXML erstellt, dies wird aber später genauer erläutert.

## 

## DTD

Bei einer DTD handelt es sich um eine Dokumenttyp-Definition (engl. Doctype-Definition). Mit Hilfe dieser können Regeln für das Dokument definiert werden, um dieses zu gestalten. DTDs dienen also als eine Art Regelwerk für das XML-Dokument und definieren die darin vorhandenen Elemente und deren Attribute.[[3]](#footnote-3)

### Notwendigkeit und Vorteile einer DTD

Eine DTD ist nicht zwingender Weise für ein XML-Dokument notwendig. So kann ein Dokument auch problemlos ohne DTD eingelesen und verarbeitet werden, dennoch bringt die Verwendung einige Vorteile mit sich:

* Ordnung / Struktur in dem Dokument: mit einer DTD erstellt man Regeln (für eine Struktur), die befolgt werden müssen, daher ist das Dokument übersichtlicher und strukturierter, kann also auch einfacher gelesen werden.
* Keine Missverständnisse bei mehreren Programmierern: mit einer DTD kann sichergestellt werden, dass alle Programmierer bei dem Erstellen oder Verarbeiten des Dokuments der gleichen Struktur folgen. Damit sind Fehler und unterschiedliche Aufbauten der Dokumente ausgeschlossen.
* Mehrere Benutzer können die Referenzstruktur verwenden: Es gibt viele XML-Applikationen, die über eine DTD verfügen (z.B. FXML, CFD uvm.). Dank dieser können verschiedene Nutzer damit arbeiten, ohne jedes Mal eine neue Sprache erstellen zu müssen.

Der Nutzen einer DTD ist also sehr groß. Allerdings sollte trotzdem überlegt werden, ob sie wirklich notwendig oder sinnvoll ist. Wenn das XML-Dokument größer als die DTD wäre, ist es sicher eine bessere Wahl, auf sie zu verzichten.

### Ein Bild, das Elektronik, Computer, Laptop, drinnen enthält. Mit hoher Zuverlässigkeit generierte BeschreibungBeispiel einer DTD (Auszug aus der CalML)

Abbildung : Ausschnitt aus CalML

Bei CalML handelt es sich um die für das Projekt erstellte DTD. Sie dient zur Speicherung der Kalenderdaten und legt die Regeln für das verwendete XML-Dokument zur Speicherung der Kalenderdaten fest. Bei dem rechts abgebildeten Quelltext sollte beachtet werden, dass es sich lediglich um einen Ausschnitt aus der gesamten DTD handelt, da das Beispiel sonst zu komplex wäre.

Abbildung 2: Ausschnitt aus CalML DTD

Mit der rechts abgebildeten DTD (Abb.1) ist das Speichern von Terminen verschiedener Personen möglich. Hier fallen die Schlüsselwörter !DOCTYPE, !ELEMENT und !ATTLIST sofort ins Auge:

Mit dem Schlüsselwort !DOCTYPE wird ausgesagt, dass es sich bei mit den „[]“-Klammern umschlossenen Bereich um die DTD handelt. Würde man das XML-Dokument jetzt übersetzen lassen, würde das Programm zum Einlesen der Datei diesen Teil nicht ausgeben.

!ELEMENT dient zur Definition eines Elements, das später im XML-Dokument benutzt werden kann. Der Bezeichner innerhalb der Klammer weist auf Unterelemente des beschriebenen Typs hin. In diesem Fall bedeutet das, dass in der Dokumenthierachie nur day-Elemente unter einem month-Element stehen dürfen (Abb. 2). Die Quantität dieser Unterelemente kann über einen folgenden Operator festgelegt werden: in diesem Fall liegt die Anzahl der möglichen dieser Unterelemente zwischen null und theoretisch unendlich (gekennzeichnet durch den \*-Operator), wäre der physische Speicher nicht begrenzt.

Ein Bild, das Elektronik, Computer enthält.

Mit hoher Zuverlässigkeit generierte Beschreibung

Abbildung 2: Beispiel Ausschnitt XML-Dokument

Mit !ATTLIST ist es möglich, einem Element Attribute hinzuzufügen. Hierfür gibt man den Namen des Elements an, gefolgt von den Namen des Attributs, den Datentyp und einen Standardwert. Auf den oben abgebildeten Fall bezogen (Abb. 1), ist der Datentyp CDATA, (ausgeschrieben Character Data) mit einer Zeichenkette vergleichbar ist. Der Standardwert ist #REQURIED, was bedeutet, dass er immer angegeben werden muss und dieses Attribut nicht optional ist. Ein optionales Attribut hätte als Standartwert #IMPLIED.

# Terminkalender

Im folgenden Abschnitt soll die Entstehung, die Funktionen und die dafür geschriebene Software erläutert werden. Eine kleine, kürze Dokumentation ist in dem GitHub-Repository[[4]](#footnote-4) zu finden. Hier werden alle Funktionen und Attribute kurz und bündig erklärt, um das Verständnis beim Lesen des Codes zu erhöhen. Jedoch erfüllt die Seminararbeit den gleichen Zweck, nur ausführlicher.

## Produkt und Funktionen

## Beginn des Projekts

Die Planung ist bei einem Projekt dieser Größe maßgeblich. Neben der Planung der Programmierung und des Designs dürfen auf keinen Fall die Wünsche der Auftraggeber, Nutzer oder Kunden vernachlässigt werden.

Über die Aufgabenstellung beispielsweise wird eine persistente Datenspeicherung über XML verlangt. In Verbindung damit hat sich die verantwortliche Lehrkraft eine DTD gewünscht, damit das Projekt auf seiner eigenen Auszeichnungssprache basiert.

### User Stories

Da ein realer Auftraggeber mit Wünschen in diesem Sinne nicht vorhanden ist, sind die Funktionen aus Wünschen von Freunden und Verwandten entstanden. Bei dieser Umfrage sind folgende Ergebnisse hervorgegangen:

Für die gefragten Personen war es besonders wichtig, dass jeder seine Freizeit nutzten kann und diese somit fest in die Planung der Termine eingebunden ist. Die Möglichkeit der Buchung von wöchentlichen Freizeitphasen oder spezifischen Zeiten als Freizeit festzulegen ermöglicht dies. Soll dieser Termin jetzt von einem anderen Nutzer belegt werden, so wird dieser informiert, dass der aktuelle Bereich bereits belegt ist und er sich bitte an diesen Nutzer wenden möchte.

Eine weitere Funktion, die in den meisten Kalenderprogrammen Standard ist, sind Permanenttermine. So wird das Eintragen von sich täglich, wöchentlich, monatlich oder jährlich wiederholenden Terminen deutlich einfacher gemacht und damit die Nutzerfreundlichkeit gesteigert.

### Tasks und Prototypen

### Zeitliche Planung

Die zeitliche Planung der Seminararbeit fand im Rahmen der zweiten Zwischenaufgabe im W-Seminar statt. Hier war genau dies gewünscht. Die Verfasser sollten, die von ihnen bereits aufgestellten Prototypen und Meilensteine mit Deadlines versehen. Das Ergebnis dieser Aufgabe war folgendes[[5]](#footnote-5):

Es ist zu erkennen, dass die zeitliche Planung einige Zeit vor dem Abgabetermin der Seminararbeit endet. Diese Zeitspanne dient als „Puffer“. Für den Fall, dass mal eine Klausur, oder andere wichtige Termine Priorität haben und daher nicht an der Seminararbeit gearbeitet werden kann, bleibt immer noch genug Zeit, um die noch ausstehende Arbeit nachzuholen. Dies soll dienen um Stress zu vermeiden.

## Zusammenarbeit der Klassen (Klassendiagramm)

Bei einem so großen Projekt mit vielen Klassen und Referenzen ist es sehr wichtig, sich zu überlegen, nach welchem System man seine Klassen ordnen möchte. In diesem Fall wurde das XYZ -System verwendet. Hier gibt es drei Bereiche: die View also die Visualisierung der Inhalte mit Hilfe einer GUI (Graphical User Interface /Benutzeroberfläche), die Logik also die logischen Abläufe und die Verarbeitung und Auswertung der Daten, die über GUI oder die Datenspeicherung kommen und XYZ.

## DTD (eigene Sprache)

Die für das Projekt verwendete Sprache heißt CalML (Calendar Markup Language) und dient zu Speicherung der Termine und anderer Informationen, die für den Kalender notwendig sind.

### Struktur

Die DTD besteht im Wesentlichen aus drei Teilen:

1. Die Speicherung der Nutzer: Hier wird zum einen gespeichert, wer der aktuelle Nutzer ist, aber auch welche Nutzer generell in dem System gespeichert sein sollen. Dadurch wird ein schneller Zugriff auf die Namen der Nutzer ermöglicht, was besonders für die Datenverarbeitung sehr wichtig ist.
2. Die Permanenttermine: Hier werden Termine gespeichert, die sich täglich, wöchentlich, monatlich oder jährlich wiederholen. Der Grund, für die getrennte Speicherung von den normalen/ einmaligen Terminen, liegt darin, dass so die Speicherung deutlich einfacher und effizienter ist. Der Termin muss nur einmal abgespeichert werden und nicht je nach Wiederholung einmal pro Tag bis einmal pro Jahr. Außerdem ist so die verwendete Rechenleistung beim Finden eines Termin deutlich geringer, als wenn er bei den anderen Terminen gespeichert wäre.
3. Speicherung der normalen /einmaligen Termine: Um das Suchen nach einem bestimmten Termin zu vereinfachen und eine geordnete Struktur in die DTD zu bringen, sind diese erst nach Jahr, anschließend nach Monat und zuletzt nach Tag unterteilt. Die Tagelemente können nun Perioden als Child-Elementen haben, in denen die Start- und Enduhrzeiten der Termine, sowie Inhalte gespeichert werden. Um die Größe der Datei zu minimieren, werden Jahre, Monate und Tage nur dann erzeugt, wenn eine Periode in diesem angelegt werden soll. Existiert ein Tag-, Monat- oder Jahreselement bereits, so erstellt das Programm nicht extra einen neuen Node, sondern speichert den Termin in den bereits vorhandenen.

## Schnittstelle Java XML (Datenspeicherung)

Als Schnittstelle zwischen den Java-Klassen und dem XML-Dokument dient eine Bibliothek, die sich DOM nennt.

### Wahl der verwendeten Bibliothek

Zur Auswahl einer Schnittstelle Java-XML boten sich verschiedene Möglichkeiten: JAXB[[6]](#footnote-6), SAX und DOM[[7]](#footnote-7). Im Folgenden ist eine Tabelle zu sehen, in der die jeweiligen Vor- und Nachteile der verschiedenen Bibliotheken erläutert werden.

|  |  |  |
| --- | --- | --- |
|  | Vorteile | Nachteile |
| SAX | Performanter als DOM | Nicht standarisiert |
|  |  | Unflexibler Zugriff auf Elemente (nur erstes und letztes Element direkt auslesbar) |
| DOM | Flexibler Zugriff auf alle Elemente | Gesamtes XML-Dokument wird in den Hauptspeicher geladen |
|  | Direkter Zugriff über getElementById() oder getElementByTagName()auf Elemente | Zugriff über Knotenhierachie ist teilweise sehr kompliziert |
|  | Modifikation des Dokuments sehr einfach |  |
|  | Standarisiert |  |
| JAXB | Ganze Objektzustände können gespeichert werden |  |
|  | Sehr performant |  |

Wie zu erkennen ist, sticht JAXB als sehr positiv heraus. Die Geschwindigkeit der Datenverarbeiten von JAXB liegt weit über der von DOM, wie in der obenstehenden Tabelle zu erkennen ist, jedoch bringt DOM einige fallspezifische Vorteile. Zum einen ist es hier sehr einfach, Daten zu ändern und einzutragen, ohne ein gesamtes Objekt der Klasse zu erstellen und auch das Ändern einzelner Attribute funktioniert bei DOM deutlich einfacher. Außerdem ist die Anzahl der Nodes, die eingelesen oder geschrieben werden, von so geringer Größe, dass keine erkennbaren Verzögerungen auftreten und so die Benutzerfreundlichkeit und die Performance des Programmes nicht sichtbar beeinflusst werden. Dies ist auch der Grund, warum die Wahl letztendlich auf DOM und nicht auf JAXB fiel.

### Implementierung

Die Implementierung der Schnittstelle geschah über drei Klassen:

1. Die Klasse XMLReader stellt die direkte Schnittstelle zu der XML-Datei dar. Hier sind die notwendigen Imports für DOM eingebunden und hier werden alle Informationen direkt aus dem Dokument ausgelesen bzw. in das Dokument geschrieben. Die Funktionen beschränken sich hier auf das einfache Beschreiben und Lesen der Datei. Eine Überprüfung auf Vorhandensein des Nodes, oder der überstehenden Nodes (Parents) findet nicht statt. Auch beim Beschreiben der Datei gibt es keine Prüfung dafür, dass eine solcher Node schon existiert.

Diese Aufgaben werden in den Klassen Reader und Writer übernommen.

1. Die Klasse Reader hat die Aufgabe, die in den Nodes gespeicherten Informationen in einen String zu übersetzen. Hierfür wird der entsprechende Node in ein Element umgewandelt. Mit Hilfe von Methoden der DOM-Bibliothek ist es nun möglich die Informationen aus dem Node in Form eines Strings zu bekommen.
2. Die Klasse Writer übersetzt die String-Eingabe (des Nutzers über die GUI) in Nodes. Das bedeutet, dass die Funktionen dieser Klasse über die Eingabeparameter Attributwerte und Inhalte für die Elemente erhalten. Diese werden dann dem neuen Node hinzugefügt, der dann an den XMLReader weitergegeben wird. Der XMLReader schreibt diese anschließend in das XML-Dokument.

Die Schnittstelle zwischen Java und XML bzw. Logik (Klasse Logic) und XML ist nach diesem Konzept aufgebaut, da in der objektorientierten Programmierung jede Klasse ihre eigene Funktion haben soll. Was auch bedeutet, dass jede Klasse nur eine Aufgabe haben soll. So werden die Klassen kleiner gehalten und der Quellcode der Funktionen übersichtlicher. Ein weiter Vorteil hier ist, dass Funktionen so an verschiedenen Stellen verwendet werden können, da sie zwingend für nur einen Zweck ausgelegt sind, der dadurch aber auch weniger spezifisch ist .

## Logic

Die Logic-Klasse dient, wie der Name schon sagt, für die Logik des Projektes. Sie verarbeitet die aus der GUI übergebenen Daten. Außerdem bündelt sie Informationen, um diese dann an die Benutzeroberfläche zu übergeben. Der Ursprung der Klasse liegt in der objektorientierten Programmierung. Die Entscheidung viel aktiv gegen die Aufteilung der Funktionen auf die Klassen Reader und Writer, da diese bereits eine andere Aufgabe haben und auch einige Nachteile dadurch entstünden. Hätten sie mehrere Aufgaben, würde das der objektorientierten Programmierung widersprechen. Ein weiterer Vorteil dieser Klassenstruktur ist, dass es sehr einfach möglich ist zu überprüfen, ob ein Termin bereits vorhanden ist, bevor es zum Eintragen kommt. Da es dafür aber notwendig ist, eine Referenz auf ein Reader-Objekt und ein Writer-Objekt zu haben, wäre diese Funktion ohne die Logic-Klasse nur schwer durchsetzbar.

## GUI mit JavaFX

Bei JavaFX handelt es sich um ein Framework zur Erstellung von Java-Applikationen. Hiermit ist das Erstellen von Benutzeroberflächen (GUIs) für verschiedene Plattformen[[8]](#footnote-8) möglich. Es wurde entwickelt, um eine neue Alternative zu AWT und Swift zu liefern, die bis 2014 als Standardlösungen für grafische Anwendungen galten, jedoch einige Mängel besonders im Bereich Medien und Animationen aufwiesen und dient seither als Standard für die Entwicklung für GUIs.

### Entscheidung über Methode zur Erstellung der GUI

Es gibt verschiedene Möglichkeiten, eine GUI mit JavaFX zu erstellen.

1. Der wahrscheinlich einfachste Weg ist mit der Applikation SceneBuilder[[9]](#footnote-9). Hier werden die GUIs über eine Benutzeroberfläche erstellt, die das Verschieben und Positionieren einzelner Elemente sehr initiativ ermöglicht. Bei der Arbeit mit diesem Programm traten jedoch einige Fehler auf. Die Größenverhältnisse von Elementen und Fenster wurden bei der Arbeit mit nicht Standardbildschirmauflösungen verzogen und nicht eingehalten. So waren die Elemente viel zu groß für das Fenster, obwohl die Vorschau im Programm selbst das gewünschte Ergebnis zeigte. Dies ist besonders dann ein Problem, wenn die Entwicklung der Benutzeroberfläche auf einem Monitor mit einer nicht-Standardauflösung (3:2) erfolgt. Daher wurde diese Möglichkeit verworfen.
2. Als Alternative boten sich FXML-Dateien an. Dies hat zwei Gründe: Zum einen wurden diese bereits über den SceneBuilder erstellt und es mussten lediglich die Größenverhältnisse angepasst werden. Zum anderen ist FXML aus XML abgeleitet und passt daher sehr gut in das Thema der Seminararbeit hinein.
3. Eine weitere Alternative bietet das Erstellen der Nutzeroberfläche über…

### Implementierung

Die Einbindung der FXML-Dateien in das Projekt fand über sogenannte Controllerklassen statt[[10]](#footnote-10). Diese dienen dazu, die Daten, die über die Nutzereingabe erzeugt werden, zu erfassen oder Ausgaben über die GUI festzulegen.

Die Imports import javafx.scene.\* und import javafx.fxml.\* binden die benötigten Bibliotheken in die Controllerklassen ein. Der erste Import ermöglich das Ändern der aktuellen Szene. Hierzu lädt die Controllerklasse die entsprechende FXML-Datei ein und zeigt dann eine neue Scene als Fenster. Das aktuelle Fenster wird dabei geschlossen auch temporär in den Controllerklassen gespeicherte Variablen werden hierbei verworfen, was jedoch kein Problem für die fallspezifische Programmierung darstellt. Der zweite Import dient der Kommunikation zwischen der FXML-Datei, die dass Aussehen der GUI beschreibt, und ihrer Controllerklasse. Mit @FXML wird gekennzeichnet, dass das folgende Attribute/ Objekt oder die folgende Methode in der FXML-Datei verankert sind. Die Controllerklasse definiert die Methodenrümpfe, die in der FXML-Datei deklariert werden und kann Methoden der Objekte aufrufen, die ebenfalls in der FXML-Datei deklariert werden. So kann zum Beispiel die Reaktion auf das Drücken eines Buttons in der Controllerklasse implementiert *(oder eher dokumentiert?)* werden.

Da die Eingaben des Nutzers leider nicht immer so formuliert sind, dass das Programm sie verarbeiten kann, wird eine Fehlermeldung ausgegeben, wenn falsche Formulierungen oder Uneindeutigkeiten auftreten, die Implementierung dieser Überprüfung steht jedoch in der Logic.

# Einschätzung des Projekts (Fazit)

## Einschätzung des Projekts

## Einschätzung des Produkts

# Literaturverzeichnis

1. Aus „XML Standards“ [Ha10] [↑](#footnote-ref-1)
2. <http://webkompetenz.wikidot.com/wp:auszeichnungs-programmiersprachen>, [Mü09] [↑](#footnote-ref-2)
3. Aus „XML für Dummies“, [TB02] [↑](#footnote-ref-3)
4. GitHub-Wiki: <https://github.com/Reiswaffl/Kalendar/wiki> [Br19] [↑](#footnote-ref-4)
5. Trelloboard zur zeitlichen Planung: <https://trello.com/b/Hg1mSbSR/w-seminar-familienkallender-eric-brendel> [Br19] [↑](#footnote-ref-5)
6. <https://docs.oracle.com/javase/tutorial/jaxp/dom/when.html> [↑](#footnote-ref-6)
7. SAX und DOM in Tabelle aus „XML Standards“, [Ha10] [↑](#footnote-ref-7)
8. Windows, macOs und Linux [↑](#footnote-ref-8)
9. https://www.oracle.com/technetwork/java/javase/downloads/javafxscenebuilder-info-2157684.html [↑](#footnote-ref-9)
10. YouTube- Tutorial [Sh15]

    www.youtube.com/watch?v=fWwaWCkh9Ig&list=PLNIWFrcmR15Jdevz2v30YipAJq0hnh2xM [↑](#footnote-ref-10)